



AtomVM

# Updates from the **AtomVM** Landscape

# About me (Davide Bettio)



<https://github.com/bettio/> | [davide@uninstall.it](mailto:davide@uninstall.it) |  
<https://uninstall.it/>

- From Padova (Venice area), Italy
- Long-time open-source dev (since ~2005 contributed to KDE Plasma and others).
- Fell in love with Elixir in 2017 while creating Astarte Platform.
- Started the AtomVM project in 2017
- I love hiking!

← Me combining hiking and Elixir by implementing the Meshtastic protocol in Elixir.

**What's *AtomVM*?**  
**(For anyone who  
missed it)**



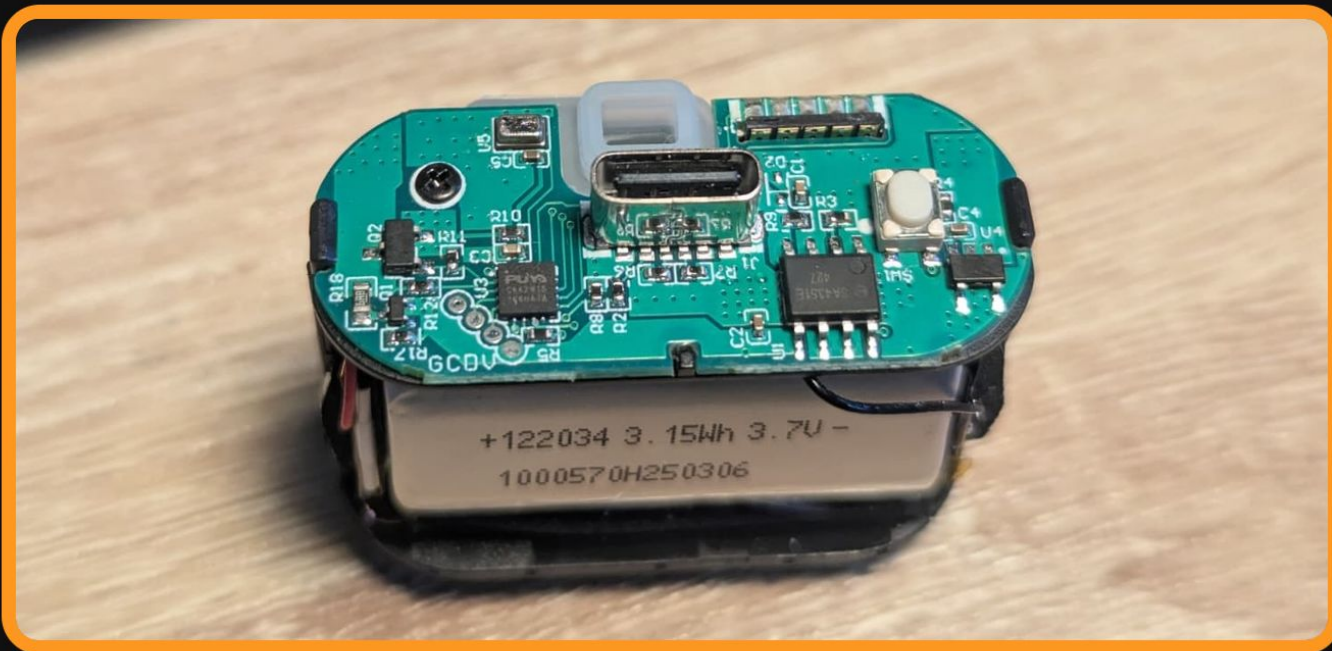
Nowadays  
Computers Are  
**Literally**  
Everywhere

# Hosting a WebSite on a Disposable Vape

---

2025-09-13 :: Bogdan Ionescu :: 6 min read (1266 words) :: [source](#) :: [report issue](#)

[#programming](#) [#arm](#) [#tools](#) [#electronics](#)



# Is There Really a Computer in There?

**It may not do much<sup>1</sup>, but yes, definitely.**

- ✓ It has a 24 MHz CPU
- ✓ It has 3 kB of SRAM memory
- ✓ It has 24 kB of flash storage
- ✓ It can do I/O

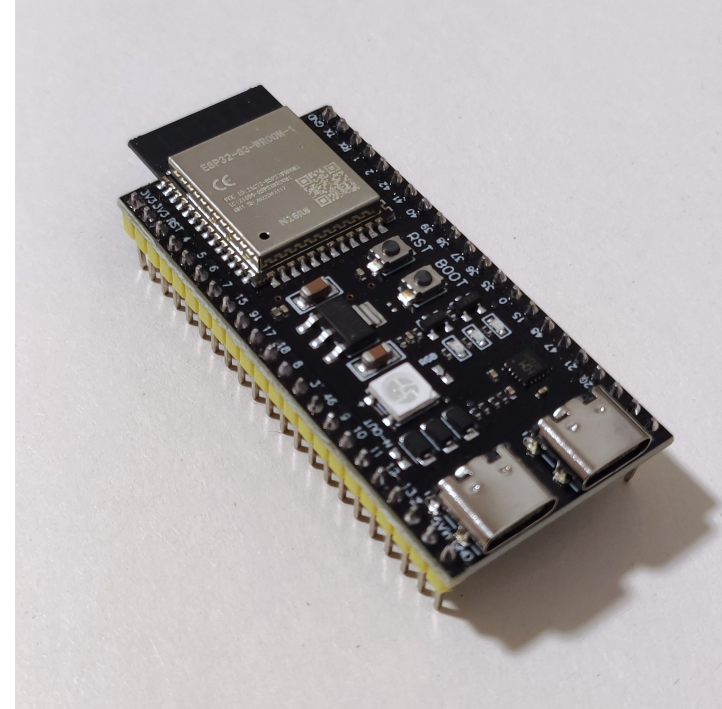
<sup>1</sup> To be fair, we went to the Moon with a computer roughly comparable to this.



Let's Focus on  
Something  
More Capable

# An ESP32

- ✓ Cost < 5 €
- ✓ Connectivity: Wi-Fi, Bluetooth, etc.
- ✓ Lots of GPIOs & integrated peripherals
- ✓ Low Power / Battery-friendly



## Also an ESP32

- ⚠ Dual Core BUT at 240 MHz
- ⚠ RAM: ~500KB - 8MB
- ⚠ Flash: 4MB - 16MB
- ⚠ No advanced operating system

**This is not enough for the BEAM.**

**No BEAM, No Elixir?**



<https://www.reddit.com/r/PallasCats/comments/1d8j3jd/bol/>

# AtomVM to the Rescue

**AtomVM is a lightweight virtual machine, implemented from scratch, designed to run compiled Erlang, Elixir and Gleam code on microcontrollers with limited resources.**

Goal: Compatible with the BEAM

- Key Trade-offs:
  - Memory First: RAM & Flash are precious
  - Portability: New targets in hours, not months
  - Flexible Requirements: Adaptable core

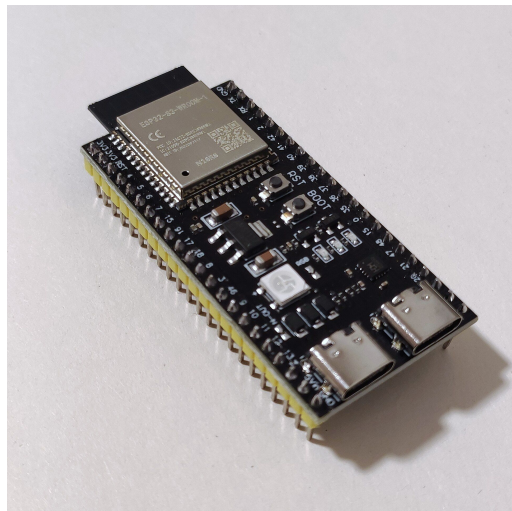
# The BEAM Ecosystem Spectrum



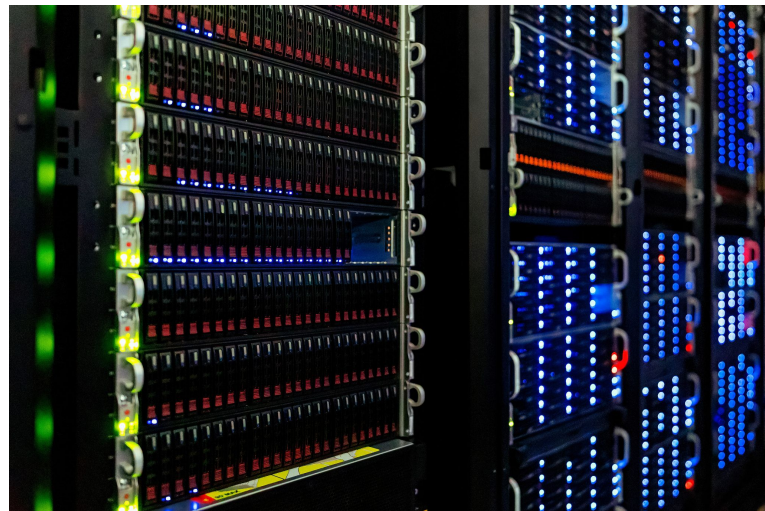
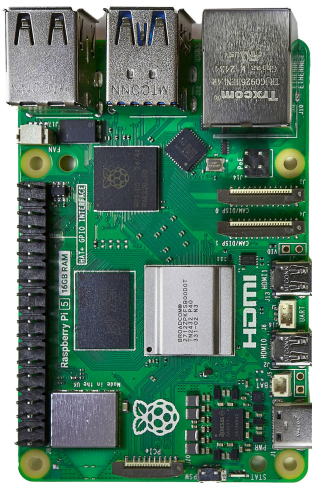
elixir



AtomVM



BEAM

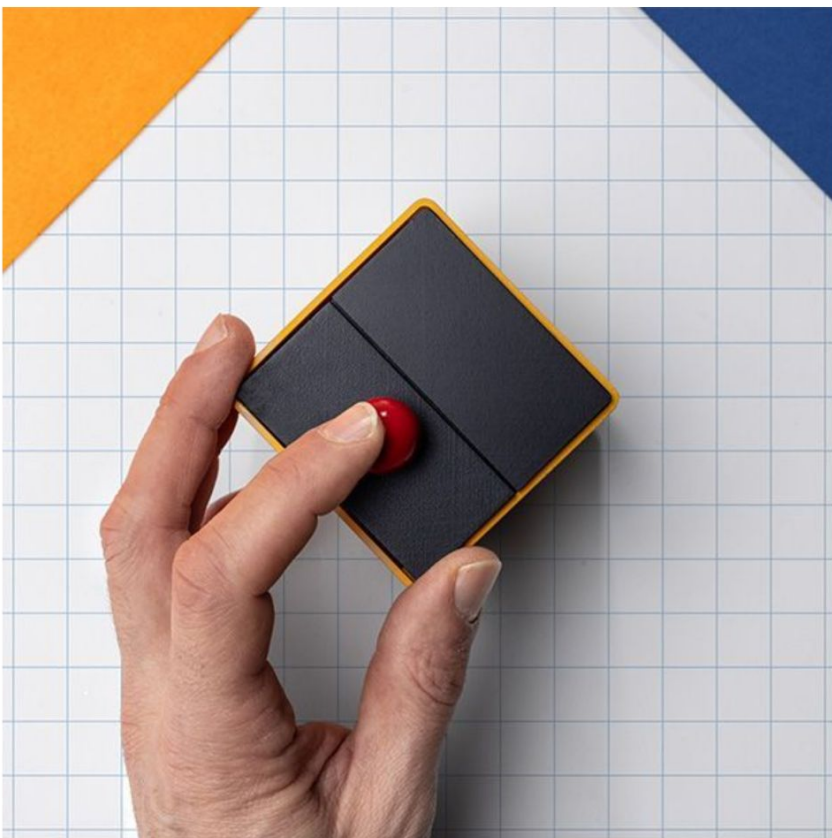




# Is AtomVM Ready For Production?

Source:

<https://www.flickr.com/photos/tambako/31556104335/in/photostream/>



## KEY FEATURES

- Over 500 unique sound effects & reactions
- Unlimited choreography combinations - never the same twice
- Fully modular design for easy repairs & customization
- Powered by ESP32 architecture
- Completely open source software - hack it, modify it, make it yours
- Eco-friendly construction from 100% recycled materials
- Exceptional battery life: three months on a single charge

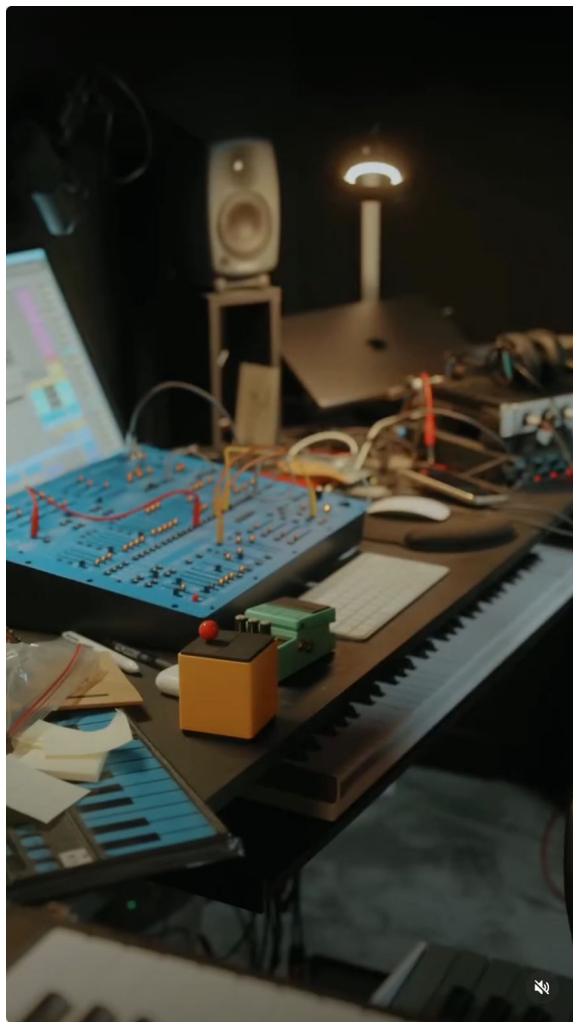
# la machine

by multipli6

The Useless Box : Reloaded



- Powered by AtomVM
- ESP32-C3
- Single-core 32-bit RISC-V CPU at 160 MHz; 400 KiB of SRAM
- 5  $\mu$ A in deep sleep!



# More than **7,000\*** la machine units running AtomVM have been shipped.

\* as of September 2026

If you have any questions, reach out to Paul Guyot. He developed the software for la machine and has made countless valuable contributions to AtomVM, including JIT support.

← A still from Jean-Michel Jarre's reel, featuring one of the la machine units.

Photo taken from: <https://manulization.com/manuls/manul-16.html>



**Something  
Cool  
Happened At  
Goatmire  
Elixir 2026**



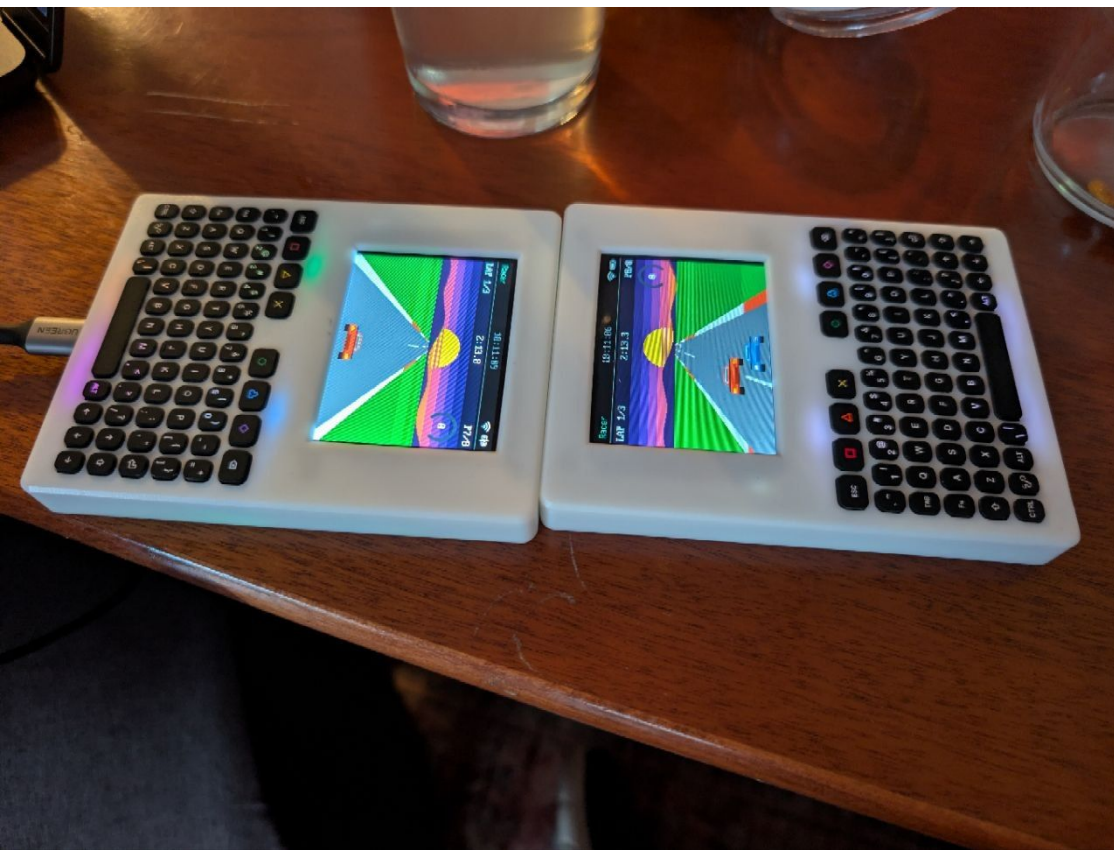
# The Goatmire 2026 Badge

ESP32, AtomVM powered  
It runs Elixir

You can:

- Exchange contact details via infrared
- Connect it to the internet and display the schedule
- Chat with other participants

And more!



← This is pure Elixir code running on a microcontroller using AtomVM.



*@haru\_manul*

**Yes,  
AtomVM is  
ready for  
production**



# AtomVM Today

# What is Supported

**AtomVM supports a carefully selected subset of BEAM and OTP features, modules and functions.**

- modules such as `lists`, `maps`, `proplists`, `sets`, `string` are available (not all functions are available but most of them are)
- processes, supervision trees and `gen_servers` are available as well
- networking APIs are available as well: `gen_tcp`, `gen_udp`, `ssl`, etc...

In other words, AtomVM implements roughly 1,600 Erlang functions.

# When an Erlang Module Is Missing

Many pure Erlang modules can run unchanged on AtomVM, provided that their dependencies and required runtime features are supported.



# Supported Data Types

## AtomVM supports the BEAM data types as well

- atoms, pids, floats, funs, references and resources.
- lists, tuples, maps
- binaries
  - Bitstrings whose total length is not a multiple of 8 bits (such as a 9-bit bitstring) are not yet supported. (WIP now)
- integers
  - including big integers, with a big caveat!

# Big Integers

**AtomVM supports big integers now**, but **why** on an embedded target?

- AtomVM previously supported only signed 64-bit integers
  - Unsigned 64-bit integers are fairly common and they were out of range
  - this problem was evident when implementing the Protocol Buffers format and decoding existing buffers
- The answer was to support integers with a larger bit width
  - But how much?
  - all big integers? just uint64? maybe int128?

# Big Integers with a 256-Bit Magnitude

- Seems like a sweet spot
- All integer operations are bounded in terms of time and memory
- The 256-bit magnitude is stored as boxed integer digits
- The sign bit is stored in the boxed integer header
- Arithmetic results outside the supported range raise an overflow error
- Logical bitwise operations never overflow, but an edge case at the 256-bit limit:

- Bitwise.bxor(-1, @int256 max) ->

[illegible]
$$(-2^{256})$$


# How the BEAM Handles Integers

Tagged term on a 64-bit target:

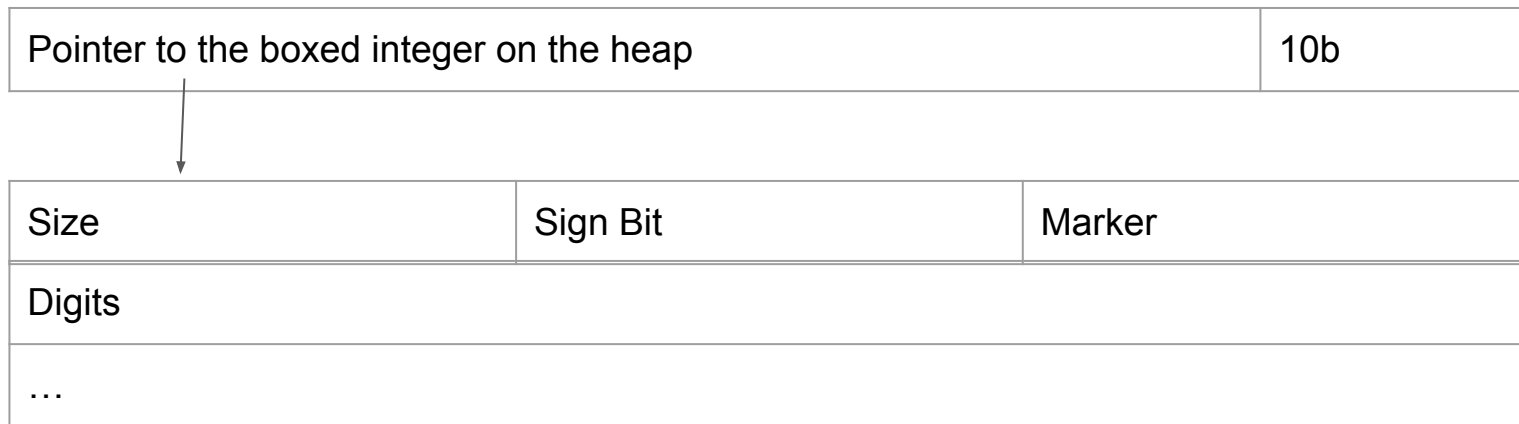
(Very efficient: it can be stored inside of a tuple, a list or a VM register without requiring any additional space!)

|                                   |       |
|-----------------------------------|-------|
| Space for a signed 60-bit integer | 1111b |
|-----------------------------------|-------|

# How the BEAM Handles Integers

Boxed term on a 64-bit target:

(boxed indirection introduces inefficiency, heap usage and GC pressure)



# Takeaway: Integer Optimization

**When the compiler knows that a term fits inside a certain range, it is able to suggest optimizations to the JIT compiler.**

```
plus_one(I) when is_integer(I, 0, 1000) ->
```

```
  I + 1.
```

The JIT compiler knows that no code for boxed integers is needed here.



# Just-in-Time and Ahead-of-Time Compilation

Worried about performance? AtomVM has optional JIT and AoT support

- Just In Time → translate to native code on the device
- Ahead of Time → translate to native code on your PC and deploy it

Supported targets: RISC-V (32-bit and 64-bit), ARMv6M and ARMv7M (32-bit), Xtensa, Aarch64, x86\_64, WebAssembly

# How Does It Work?

- AtomVM includes a translator from BEAM bytecode to native code, written entirely in Erlang
  - It can run both on the BEAM on the host and on AtomVM on the target
- With AOT compilation, native code is transparently embedded in BEAM files

The core functionality is in place; the remaining work is to integrate it into exatomvm and rebar3.

# AtomVM Performance Update

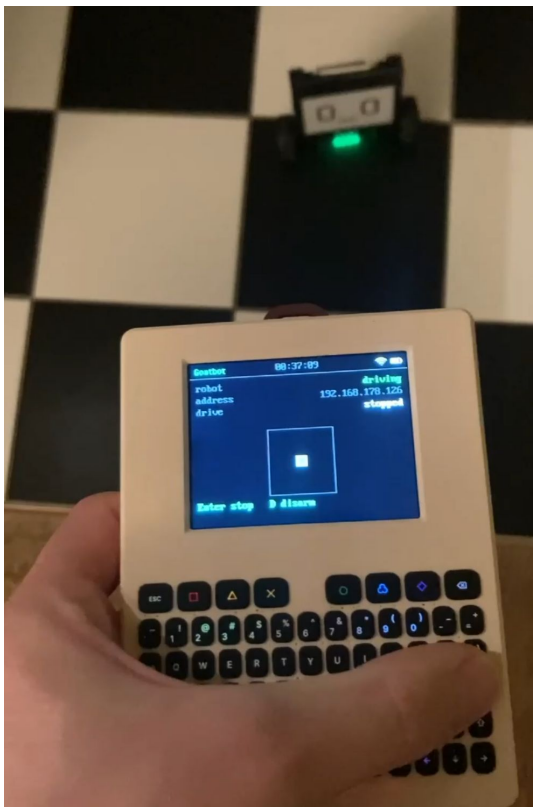
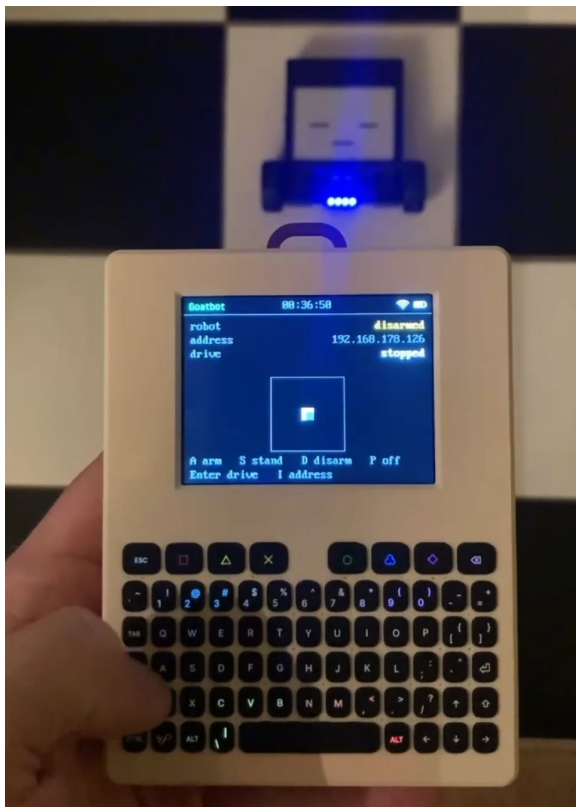
These are preliminary measurements from the v0.7.0 development branch.

**In general during v0.7.0 development we removed a number of bottlenecks and we still have some improvements in the pipeline.**

Keep in mind, AtomVM is not optimized for speed but it is **optimized for memory footprint and size**. That said:

- Some preliminary benchmarks show performance comparable to BEAM, or slightly faster in specific workloads.
- Compiling a 129-line Elixir file on AtomVM takes 209.0 ms, compared with 31.6 ms on the BEAM on the same Intel CPU.

# Erlang Distribution



AtomVM and BEAM nodes can communicate using Erlang distribution.

← A Goatmire Balance Bot running **Nerves**, **remotely controlled** by a badge running **AtomVM** via **Erlang distribution**.

See also:  
[github.com/protolux-electronics/avm\\_badge/tree/goatbot](https://github.com/protolux-electronics/avm_badge/tree/goatbot)



Photo Credit: Gus Workman

# Developer Experience Got Awesome

If you've used the AtomVM Elixir library before, it's time for a fresh look. **The new exatomvm does a lot of the heavy lifting for you.**

How to start with AtomVM now:

- Add `{:exatomvm, github: "AtomVM/exatomvm", runtime: false}` to the deps list in `mix.exs`
- `$ mix deps.get`
- `$ mix atomvm.esp32.install`
- Follow the exatomvm instructions
- `$ mix atomvm.esp32.flash`
- `$ mix atomvm.esp32.monitor`

# Gleam



# Gleam and AtomVM

- AtomVM and Gleam started around the same time. AtomVM's official tooling has historically focused on Erlang and Elixir
- **AtomVM is language-agnostic**: it executes BEAM bytecode, **including code compiled from Gleam's Erlang target**.
- **Independent community projects now provide tools and libraries** for using Gleam with AtomVM.

# Gleam + AtomVM Basic Recipe

To build software for devices with AtomVM, you need:

- Hardware APIs such as the Erlang `gpio` module or its Elixir equivalent, `GPIO`
- A build tool that helps you package all your modules and flash them to the device

# Meet Orbital

Orbital is the missing piece for building and flashing an AtomVM project with Gleam.

Getting started is straightforward:

```
gleam add --dev orbital
```

So later you can do:

```
gleam run -m orbital flash esp32 --port /dev/some_device
```

# Calling Erlang's `gpio` Module from Gleam

- Elixir developers are used to calling Erlang functions with `:module.function()`. **Gleam uses a different approach: a foreign function interface, or FFI.**
- You declare an external function and its types
- Your code stays safely typed
- The day is saved

# Using FFI in Gleam Is Easy

```
pub type GpioResult
```

```
pub type Direction {
```

```
    Input
```

```
    Output
```

```
}
```

```
@external(erlang, "gpio", "set_pin_mode")
```

```
fn set_pin_mode(pin: Int, direction: Direction) -> GpioResult
```



# Next Steps

Writing FFI bindings from scratch for every new project is repetitive and can lead to errors.

Gleam type-checks calls against the FFI declaration; the declaration must match the external function.

The alternative is using binding libraries:

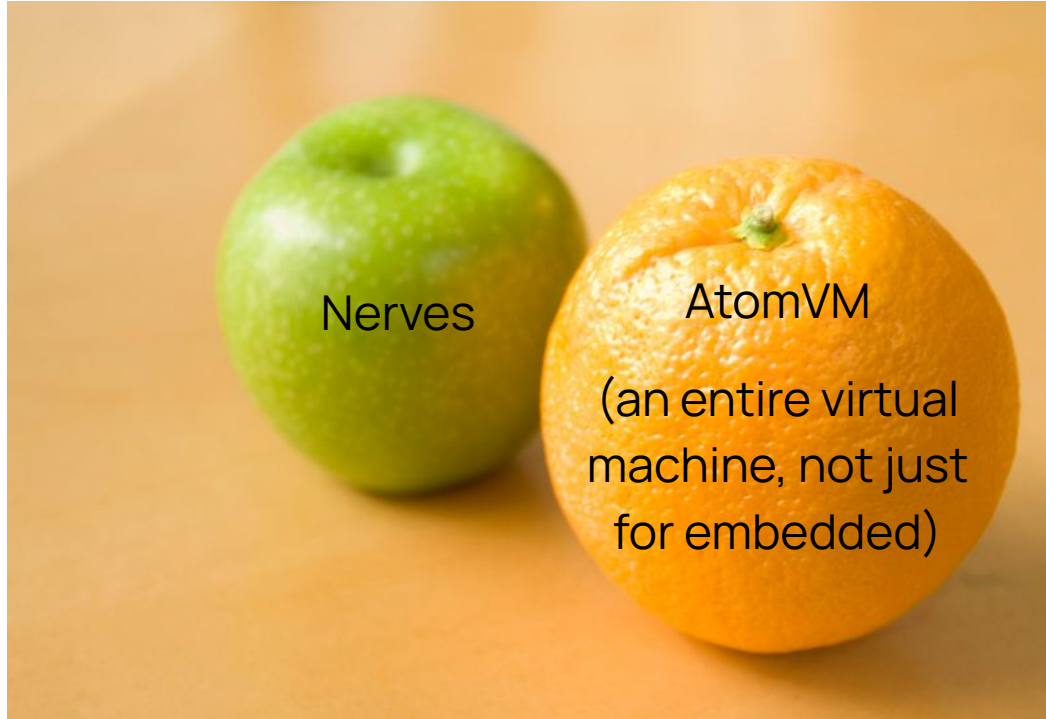
- <https://github.com/gleam-eensy/eensy>
- [https://github.com/nathanjohnson320/atomvm\\_gleam](https://github.com/nathanjohnson320/atomvm_gleam)

**Future: the long-term goal is to provide an official API that goes beyond basic bindings.**



# The Future

# Comparison with Nerves



[https://en.wikipedia.org/wiki/Apples\\_and\\_oranges#/media/File:Apple\\_and\\_Orange\\_-\\_they\\_do\\_not\\_compare.jpg](https://en.wikipedia.org/wiki/Apples_and_oranges#/media/File:Apple_and_Orange_-_they_do_not_compare.jpg)

Let's compare apples and oranges anyway.

AtomVM on MCUs, TL;DR:

- Hello world footprint: 512 KiB of flash, 32 KiB of RAM
- Perfect for cheap low power devices
- ...

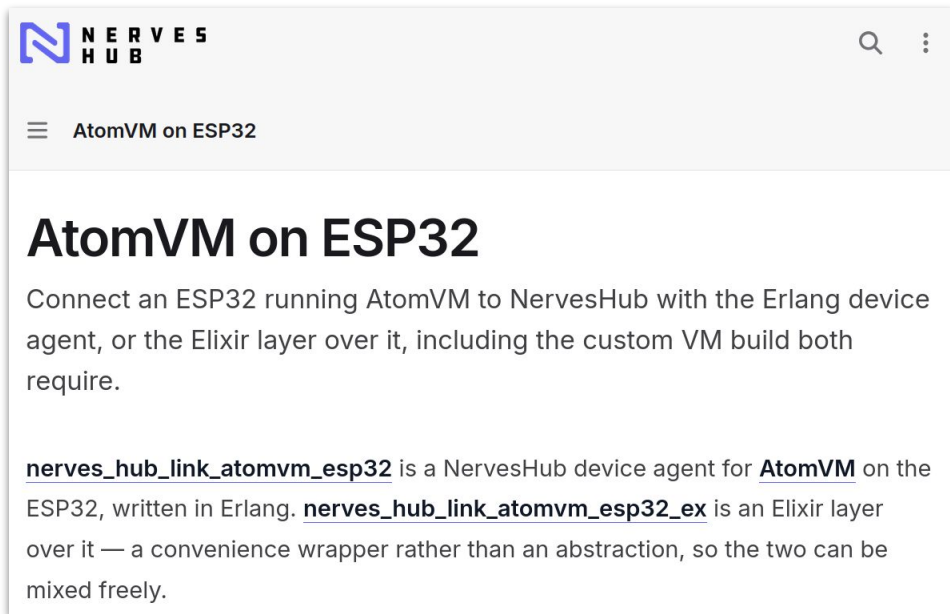
# Nerves & AtomVM

Right now, AtomVM and Nerves have very different APIs.

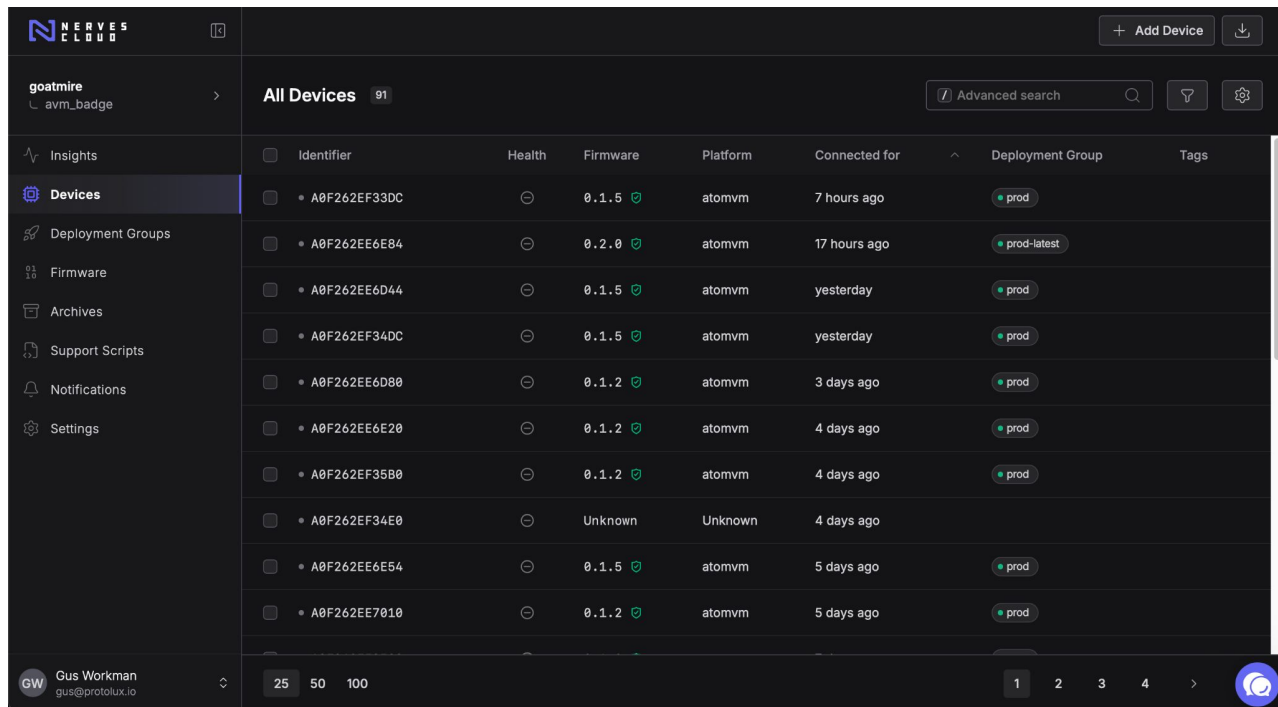
In the future, I expect Elixir applications on AtomVM to use APIs similar to those in the Nerves ecosystem, such as `Circuits.GPIO` and so on...

# NervesHub

Integration between AtomVM and the Nerves ecosystem has already started with NervesHub.



# NervesHub



The screenshot displays the NervesHub web interface. On the left is a sidebar with navigation links: Insights, **Devices**, Deployment Groups, Firmware, Archives, Support Scripts, Notifications, and Settings. The main area is titled 'All Devices' and shows a list of 91 devices. The table columns are Identifier, Health, Firmware, Platform, Connected for, Deployment Group, and Tags. The devices listed are all AtomVM instances with various identifiers, health status (green circle), firmware versions (0.1.5 or 0.1.2), and deployment groups (prod or prod-latest). The bottom of the interface shows a user profile for Gus Workman and a pagination bar with 25, 50, and 100 items per page.

| Identifier   | Health | Firmware | Platform | Connected for | Deployment Group | Tags |
|--------------|--------|----------|----------|---------------|------------------|------|
| A0F262EF33DC | ●      | 0.1.5    | atomvm   | 7 hours ago   | prod             |      |
| A0F262EE6E84 | ●      | 0.2.0    | atomvm   | 17 hours ago  | prod-latest      |      |
| A0F262EE6D44 | ●      | 0.1.5    | atomvm   | yesterday     | prod             |      |
| A0F262EF34DC | ●      | 0.1.5    | atomvm   | yesterday     | prod             |      |
| A0F262EE6D80 | ●      | 0.1.2    | atomvm   | 3 days ago    | prod             |      |
| A0F262EE6E20 | ●      | 0.1.2    | atomvm   | 4 days ago    | prod             |      |
| A0F262EF35B0 | ●      | 0.1.2    | atomvm   | 4 days ago    | prod             |      |
| A0F262EF34E0 | ●      | Unknown  | Unknown  | 4 days ago    |                  |      |
| A0F262EE6E54 | ●      | 0.1.5    | atomvm   | 5 days ago    | prod             |      |
| A0F262EE7010 | ●      | 0.1.2    | atomvm   | 5 days ago    | prod             |      |

Managing AtomVM devices and delivering application updates through NervesHub.

# More Platforms

AtomVM is not just about ESP32.

Support for platforms beyond ESP32 has improved significantly this year.

- STMicroelectronics **STM32 support has been completely revamped**
- **Raspberry Pi Pico support now covers more peripherals**

# Other Supported Platforms: Raspberry Pi Pico

- Pico 1/1W (RP2040) → **264 KiB of RAM**, dual core @ 133 MHz, 2 MiB+ flash
- Pico 2/W (RP2350) → **520 KiB of RAM**, dual core @ 150 MHz, 4 MiB+ flash
- **The 'W' models include Wi-Fi + Bluetooth**
- What's cool? Peripherals & **Programmable I/O (PIO)**

Note: Raspberry Pi Pico ≠ “classic” Raspberry Pi or Raspberry Pi Zero



# The Future: Zephyr and Other Platforms

**Version 0.7.0 is nearing release**, and a new development cycle is about to begin.

One of the goals of the new release cycle is **extending AtomVM support beyond the platforms and microcontroller families we already support**.

We would like to run AtomVM on hardware from Nordic Semiconductor or NXP.

To make this possible, **we plan to add support for Zephyr** and other real-time operating systems.



# Beyond Microcontrollers

**AtomVM is also evolving beyond constrained embedded targets.**

- **WASI:** a new port is close to landing, bringing AtomVM to standalone WebAssembly runtimes.
- **Web browsers:** the Emscripten/WASM port keeps improving, with a wasm32 JIT backend and more optimization work ahead.
- **Community ports:** new experiments are bringing AtomVM to other environments, including unikernel-style deployments for Elixir and Erlang.
- Different targets, different trade-offs: **we plan to explore optional performance-oriented optimizations** for platforms with more memory available.

Don't Wait Until  
Tomorrow:  
**AtomVM Is Ready  
For Your Next Project  
Today**

# Support this Project

Contributions of all kinds are welcome.

**Sponsorship helps** sustain maintenance and continued development.

If AtomVM is useful to you or your team,  
consider becoming a sponsor.

❤️ <https://github.com/sponsors/atomvm> ❤️





# Join Us

<https://atomvm.org/>

Discord: <https://discord.gg/QA7fNjm9Nw>

Telegram: <https://t.me/atomvm>

Documentation: <https://doc.atomvm.org/>